Data reduction and fitting

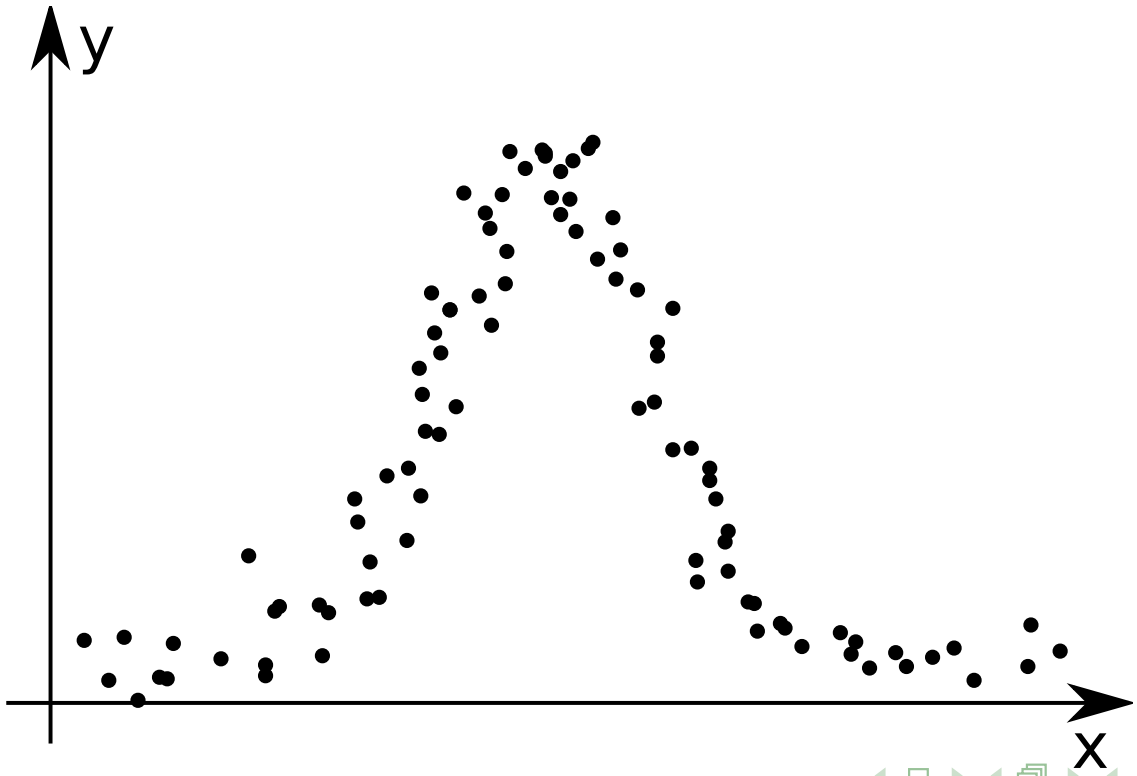
Eugeniy E. Mikhailov

The College of William & Mary

Lecture 15

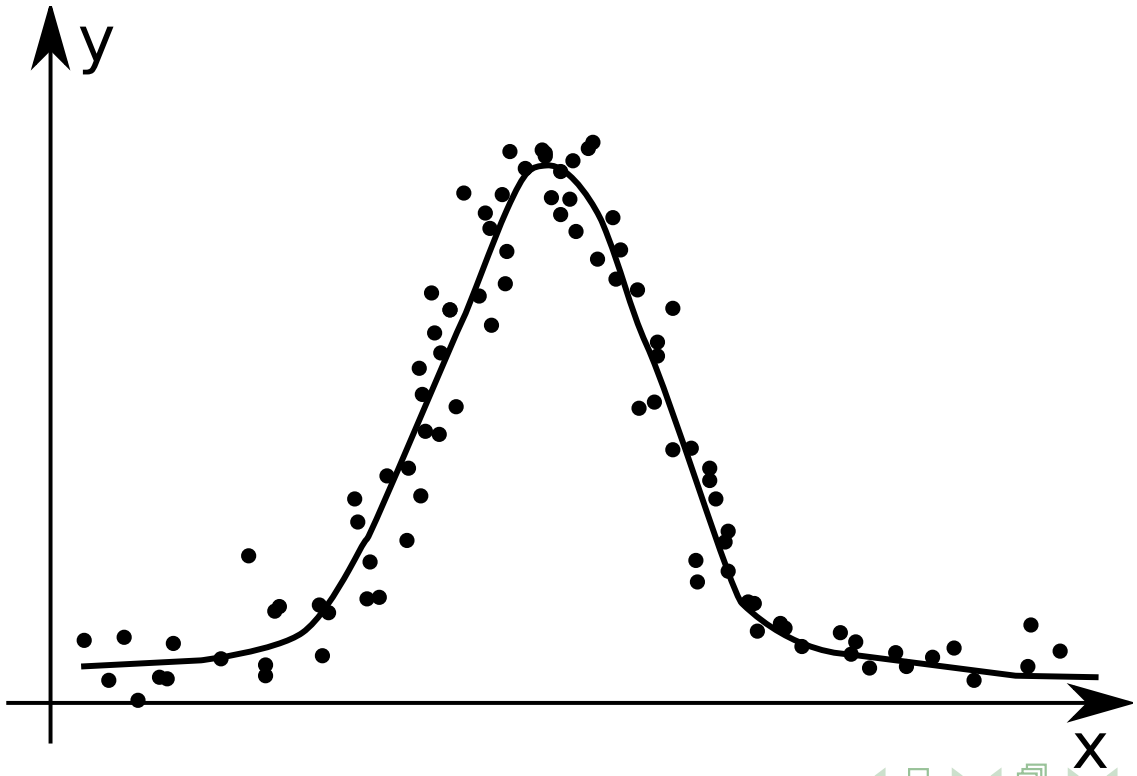
Data reduction

- Typical modern experiment generates Mega bytes or even Terra bytes of data.
- there is no way for a human to comprehend such enormous amount of data



Data reduction

- Typical modern experiment generates Mega bytes or even Terra bytes of data.
- there is no way for a human to comprehend such enormous amount of data
  - we need to post-process it and extract some important parameters
  - alternatively we want to check how our models reflect reality



Fitting

Someone measured bunch of experimental points y as a function of independent variable x. We want to extract model parameters $\vec{p}$ via fitting of the model function $f(x, \vec{p})$.

Remark: in general x and y could be vectors i.e. multi-dimensional, for example $\vec{x}$ has 2 coordinates: speed of the car and the weight of the load, and y would have the fuel consumption and the engine temperature.

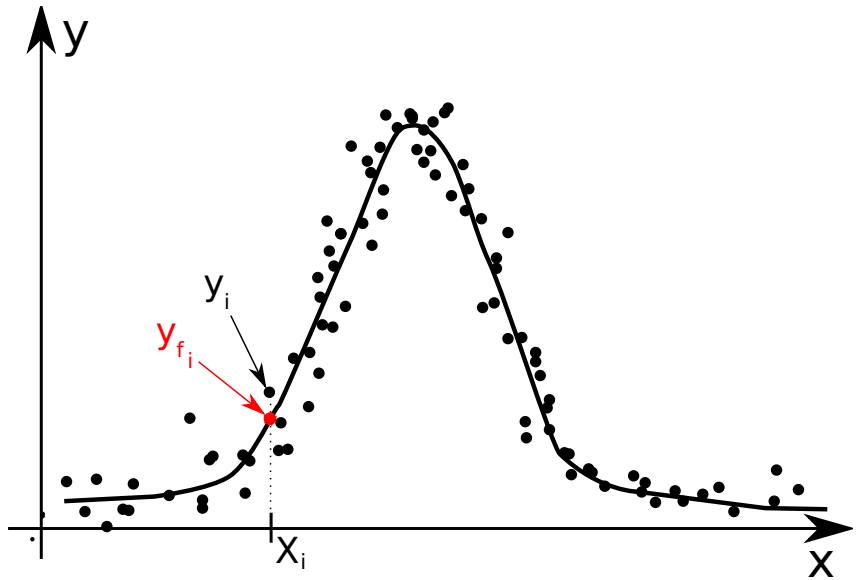
For simplicity we will focus on the one dimensional case for x and y

- we are given experimental points $x_i \rightarrow y_i$
- our model function $f(x_i, \vec{p})$: $x_i \rightarrow y_{f_i}$

Goodness of the fit

First we need to define some way to estimate goodness of the fit.

Very common is to use the sum of the squares of the fit deviations from the experiment data points.



$$\chi^2 = \sum_i (y_i - y_{f_i})^2$$

Differences of $(y_i - y_{f_i})$ are called residuals

For a given sets $\{x_i\}$, $\{y_i\}$ and f the goodness of the fit χ^2 depends only on parameters $\vec{p}$ of the model/fit function

Our job is simple: find optimal $\vec{p}$ which minimizes χ^2 using any suitable algorithm. I.e. perform so called the least square fit.

Good fit should have the following properties

- residuals should be randomly scattered around 0
  - i.e. no visible trends of residuals vs x
- standard deviation or RMS residual $= \sqrt{\frac{1}{N}\sum_i^N (y_i - y_{f_i})^2}$ should be in order of the Δy (experimental uncertainty for y)
  - the above condition is often overlooked but you should keep your eyes on it.
  - with enough fitting parameters you can make zero residuals fit but this is unphysical since all your data has uncertainties if the measurements
  - beside such fits are usually useless since any new data point usually leads to drastic modifications of the fit parameters.

Estimation of uncertainty for parameters

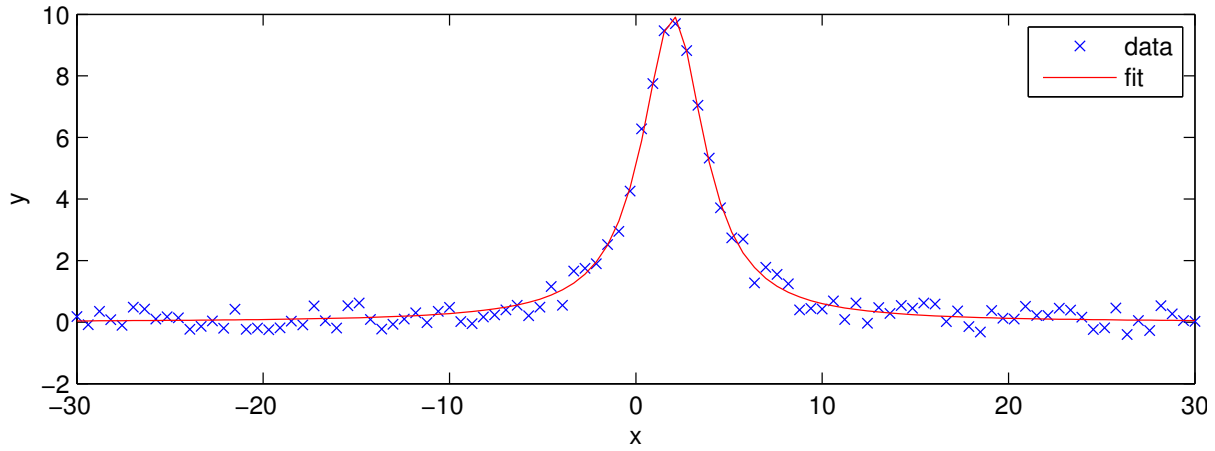
- Δp_i could be estimated by change of the χ^2,
- Δp_i: $\chi^2(p_1, p_2, p_3, \ldots p_i + \Delta p_i, \ldots) = 2\chi^2(p_1, p_2, p_3, \ldots p_i, \ldots)$

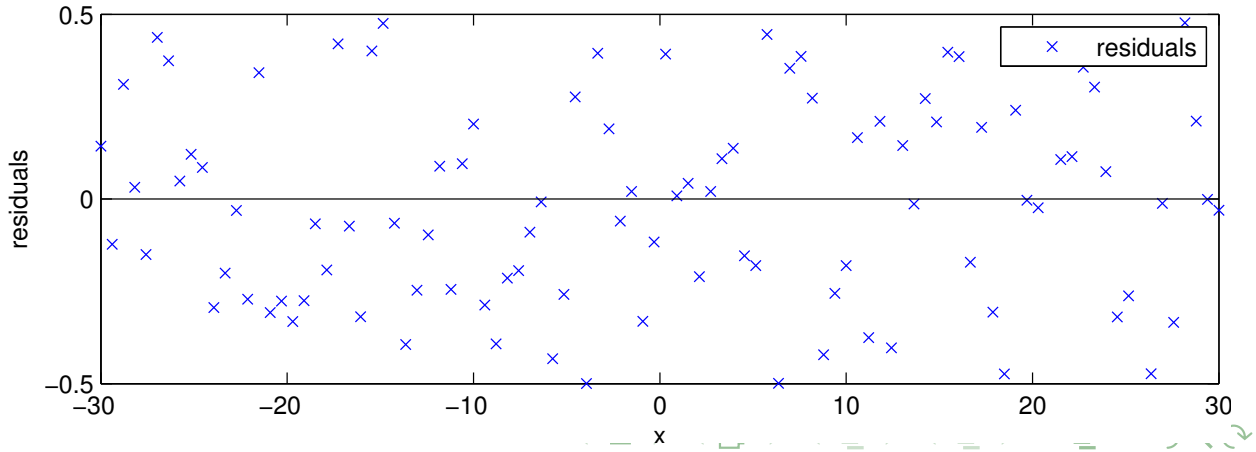
Practical realization

Have a look at 'fitter.m' where optimization of χ^2 is done with `fminsearch` matlab function.
See 'fitter_usage_example.m' for a particular usage example.

$$f(x, \vec{p}) = \frac{A}{1 + \left(\frac{x - x_o}{\gamma}\right)^2}$$

$$\vec{p} = [A, x_o, \gamma] = [9.9444, 1.9936, 2.0354]$$





Matlab built-ins

- see `fit` from the Matlab curve fitting toolbox
  - more cumbersome to start using
  - provides parameters uncertainties
- see `lsqcurvefit` from the Matlab optimization toolbox

They are faster since they take an assumption that merit function is quadratic.

Matlab built-in fit usage example

```
%% built in fit function usage example

% load initial data file
data=load('data_to_fit.dat');
x=data(:,1); % 1st column is x
y=data(:,2); % 2nd column is y

% define fitting function
% notice that it quite human readable
% Matlab automatically treat x as independent variable
f=fittype(@(A,x0,gamma, x) A ./ (1 +((x-x0)/gamma).^2) )

% let's see did Matlab guessed fit parameters right
coeffs = coeffnames(f)

% assign initial guess
pin=[3,3,1]; % [A, x0, gamma]

% We fit our data here
[fitobject,gof] = fit (x,y, f, 'StartPoint', pin)

disp('confidence interval/errorbars for  A, x0, and gamma');
ci = confint(fitobject)

builtin_fit_check(x,y, fitobject);
```